PHP, Composer et Laravel

https://github.com/heig-vd-devprodmed-course/heig-vd-devprodmed-course

Visualiser le contenu complet sur GitHub à cette adresse.

L. Delafontaine, avec l'aide de GitHub Copilot.

Plus de détails sur GitHub

Cette présentation est un résumé du contenu complet disponible sur GitHub.

Pour plus de détails, consulter le contenu complet sur GitHub ou en cliquant sur l'en-tête de ce document.

Objectifs (1)

- Décrire la différence entre l'écriture de code PHP (vanilla) et l'utilisation de librairies externes.
- Utiliser Composer pour gérer des librairies externes.
- Décrire les principes de base de Laravel.
- Lister où trouver des ressources et de l'aide sur Laravel.

Objectifs (2)

- Décrire la différence entre PHP, Composer et des frameworks comme Laravel.
- Décrire le lien entre PHP, JavaScript, Composer et npm.

PHP vanilla vs librairies externes

Qu'est-ce que PHP vanilla ?

- "PHP vanilla" fait référence à l'utilisation de PHP sans librairie ou framework externe.
- Tout le code est écrit à partir de zéro.
- Uniquement les fonctions natives offertes par PHP.

```php
<?php
echo "Hello, World!";
```

Valable pour des scripts simples ou pour apprendre les bases, mais vite limité.

Limites de tout écrire soi-même

- **Temps** : réinventer la roue pour chaque fonctionnalité.
- **Risques** : bugs et failles de sécurité.
- **Maintenance** : code à maintenir et mettre à jour soi-même.
- **Documentation** : code à documenter soi-même.

Vous l'avez expérimenté avec *"Programmation serveur 1 (ProgServ1)"* et *"Programmation serveur 2 (ProgServ2)"* : faire correctement tout soi-même est long et fastidieux.

Avantages des librairies externes (1)

- **Réutilisabilité** : code déjà écrit et testé.
- **Support** : aide(s) disponible(s) en cas de problème(s).
- **Code testé** : librairies populaires bien testées et sécurisées.
- **Documentation** : librairies souvent bien documentées.

Avantages des librairies externes (2)

- **Mises à jour** : corrections de bugs et nouvelles fonctionnalités.

Les développeur.euses peuvent se concentrer sur la création de fonctionnalités spécifiques à leur application plutôt que de réécrire des fonctionnalités de base.

Exemple : mail() vs PHPMailer

Avec la fonction native mail()

```php
<?php
$to = "destinataire@example.com";
$subject = "Sujet du message";
$message = "Ceci est le contenu du message.";
$headers = "From: expediteur@example.com";

mail($to, $subject, $message, $headers);
```

Limitations : configuration SMTP complexe, pas de pièces jointes, pas de HTML riche, emails souvent marqués comme spam.

Avec PHPMailer (1)

```php
<?php
require 'vendor/autoload.php';

use PHPMailer\PHPMailer\PHPMailer;

$mail = new PHPMailer(true);

// Configuration SMTP
$mail->isSMTP();
$mail->Host = 'smtp.example.com';
$mail->Port = 465;
$mail->SMTPAuth = true;
$mail->SMTPSecure = PHPMailer::ENCRYPTION_SMTPS;
```

```php
$mail->Username = 'utilisateur@example.com';
$mail->Password = 'mot_de_passe';
$mail->CharSet = "UTF-8";
$mail->Encoding = "base64";

// Expéditeur et destinataire
$mail->setFrom($from_email, $from_name);
$mail->addAddress('CHANGE_ME', 'CHANGE WITH YOUR NAME');

// Contenu du mail
$mail->isHTML(true);
$mail->Subject = 'Here is the subject';
$mail->Body    = 'This is the HTML message body <b>in bold!</b>';
$mail->AltBody = 'This is the body in plain text for non-HTML mail clients';

// Envoi du mail
$mail->send();
```

Avec PHPMailer (2)

Avantages :

- Support SMTP avec authentification.
- Envoi HTML riche.
- Pièces jointes.
- Gestion des erreurs
- Meilleure délivrabilité.

➡️ **Il n'a pas été nécessaire de réécrire toute la logique d'envoi d'emails : gain de temps et qualité du code augmentée.**

Composer

Qu'est-ce que Composer ? (1)

Composer est le gestionnaire de dépendances standard pour PHP. Il permet de :

- Déclarer les librairies dont votre projet dépend.
- Télécharger et installer ces librairies automatiquement.

Qu'est-ce que Composer ? (2)

- Gérer les versions et les mises à jour.
- Charger automatiquement les classes (autoloading).

Composer est à PHP ce que npm est à JavaScript, pip à Python, Maven à Java ou encore Cargo à Rust.

Concepts clés de Composer

Élément	Description
`composer.json`	Configuration du projet et liste des dépendances
`composer.lock`	Versions exactes installées (à versionner)
`vendor/`	Dossier des dépendances (ne pas versionner)
`vendor/autoload.php`	Chargement automatique des classes

Commandes de base

Commande	Description
`composer init`	Créer un nouveau `composer.json`
`composer require`	Ajouter une dépendance
`composer install`	Installer les dépendances depuis le lock
`composer update`	Mettre à jour les dépendances
`composer remove`	Supprimer une dépendance

Packagist

Packagist est le dépôt principal de paquets PHP.

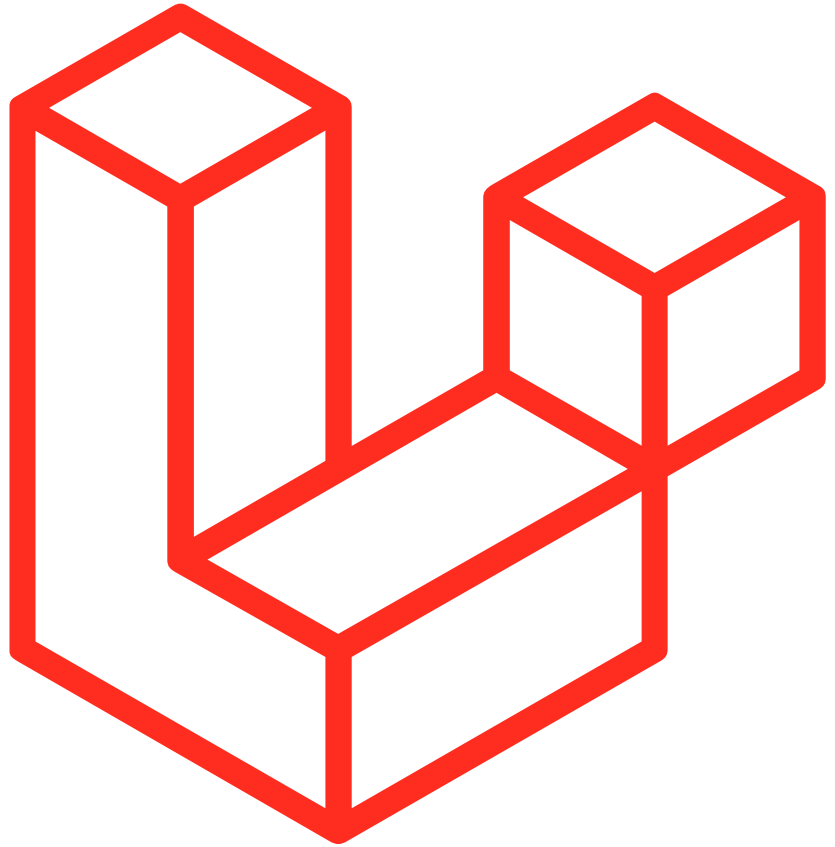
Composer utilise Packagist pour rechercher et télécharger les librairies.

Il n'est plus nécessaire de télécharger manuellement les librairies comme en ProgServ2 !

Introduction à Laravel

Qu'est-ce que Laravel ?

- Framework PHP open-source créé par Taylor Otwell (2011).
- L'un des frameworks PHP les plus populaires.
- Structure de projet organisée.
- Outils pour les tâches courantes.
- Documentation complète
- Communauté (extrêmement) active.



Pourquoi utiliser un framework ?

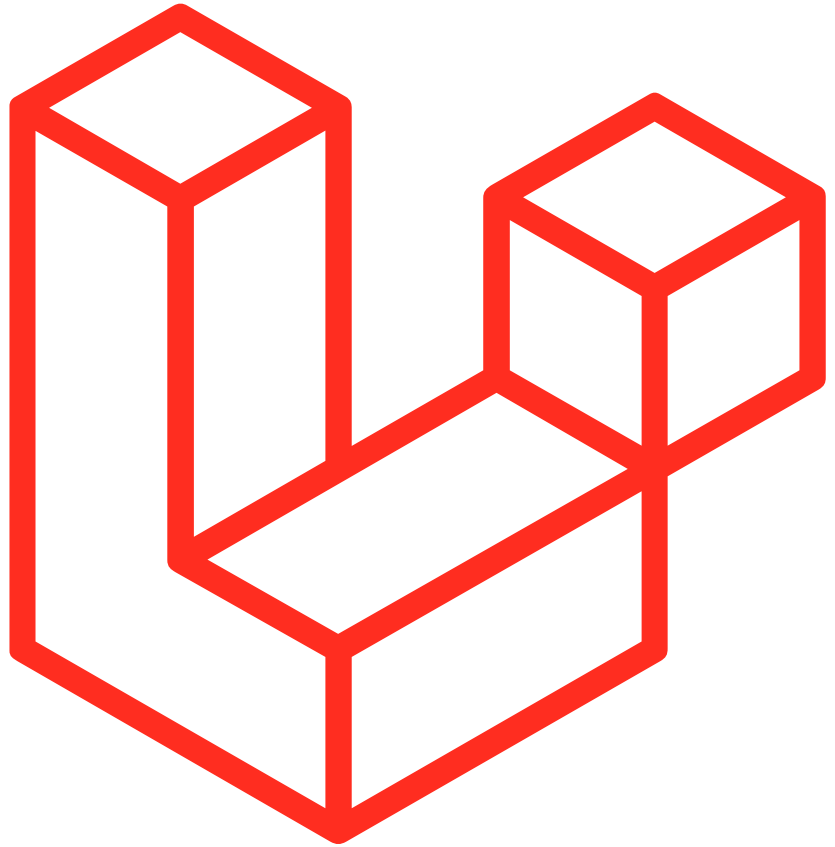
- **Productivité** : fonctionnalités communes déjà implémentées.
- **Sécurité** : protection contre les failles courantes.
- **Maintenabilité** : structure de code standardisée.
- **Communauté** : support, tutoriels, packages.
- **Bonnes pratiques** : encouragées par le framework.

Vous vous concentrez sur la logique métier de votre application plutôt que sur les détails techniques.

Principes clés

Laravel repose sur plusieurs principes clés qui facilitent le développement web :

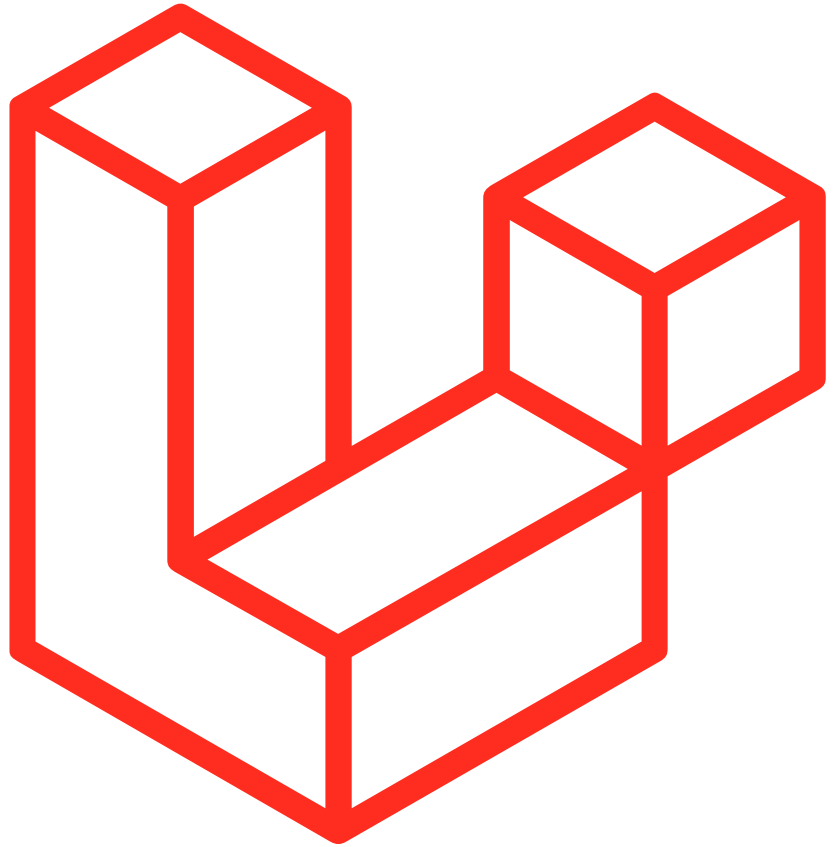
- Architecture MVC.
- Convention over configuration.
- Système de routing.



Architecture MVC

Laravel utilise l'architecture Model-View-Controller (MVC) :

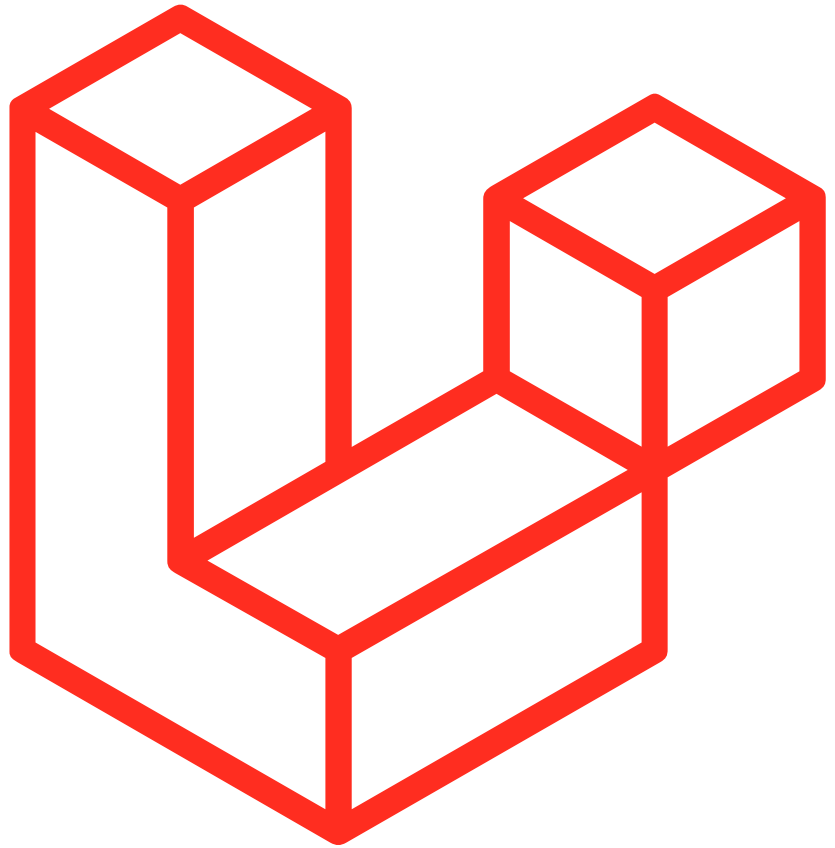
- **Model** : représente les données et la logique métier.
- **View** : affiche les données à l'utilisateur.
- **Controller** : gère les requêtes (entrées/sorties) et coordonne Model et View.



Convention over configuration

Si vous suivez les conventions de Laravel, tout fonctionne automatiquement.

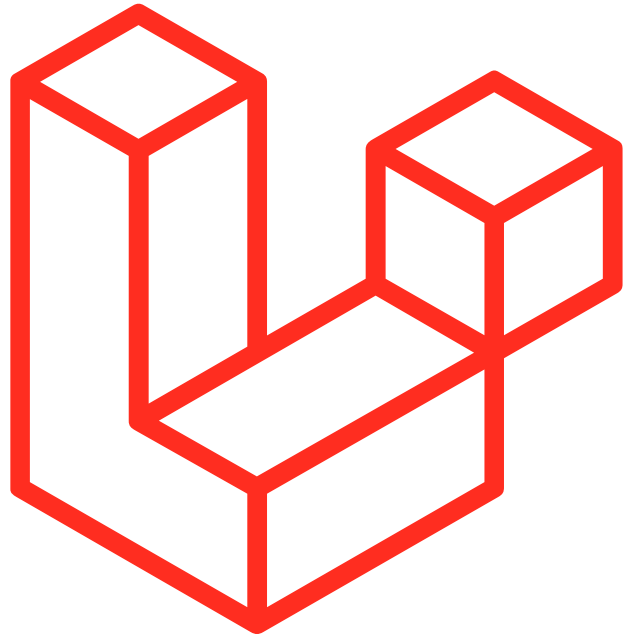
Exemple : un modèle `User` cherche automatiquement une table `users`.



Ecosystème Laravel

Outils intégrés (vus dans les prochaines séances) :

- **Artisan** : interface en ligne de commande de laravel.
- **Blade** : moteur de templates.
- **Eloquent** : ORM pour la base de données.
- **Middleware** : filtrage des requêtes HTTP.
- Et plus encore !



Ressources et aide

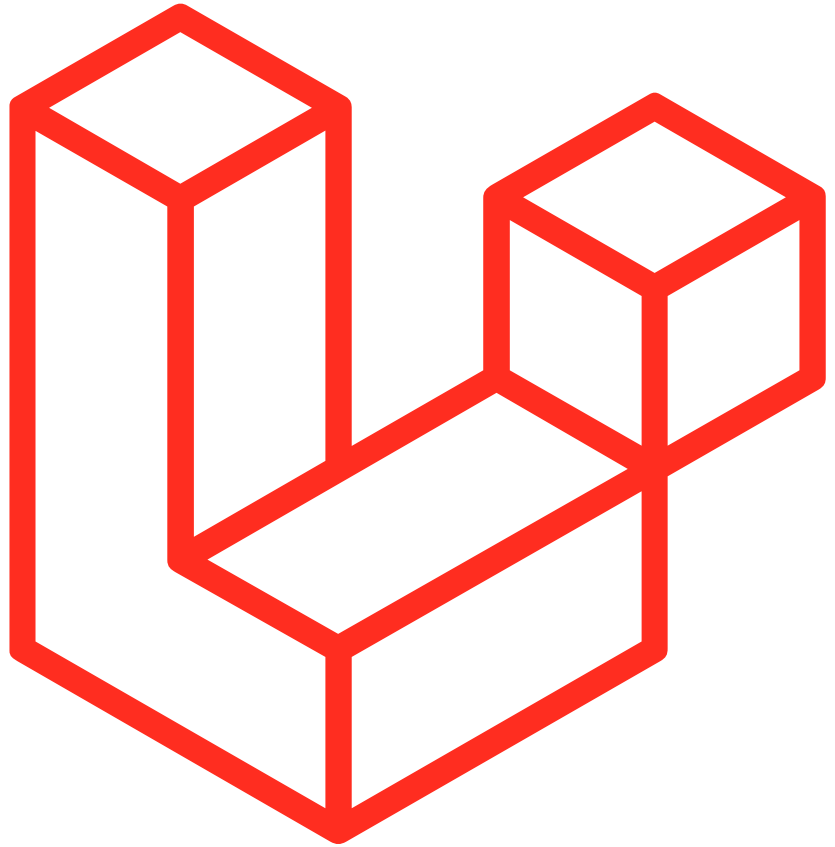
Laravel dispose d'un riche écosystème de ressources pour apprendre et obtenir de l'aide :

- Documentation officielle.
- Communauté active.
- Dépôts GitHub.

Durant le cours, ces différentes ressources seront utilisées pour approfondir vos connaissances.

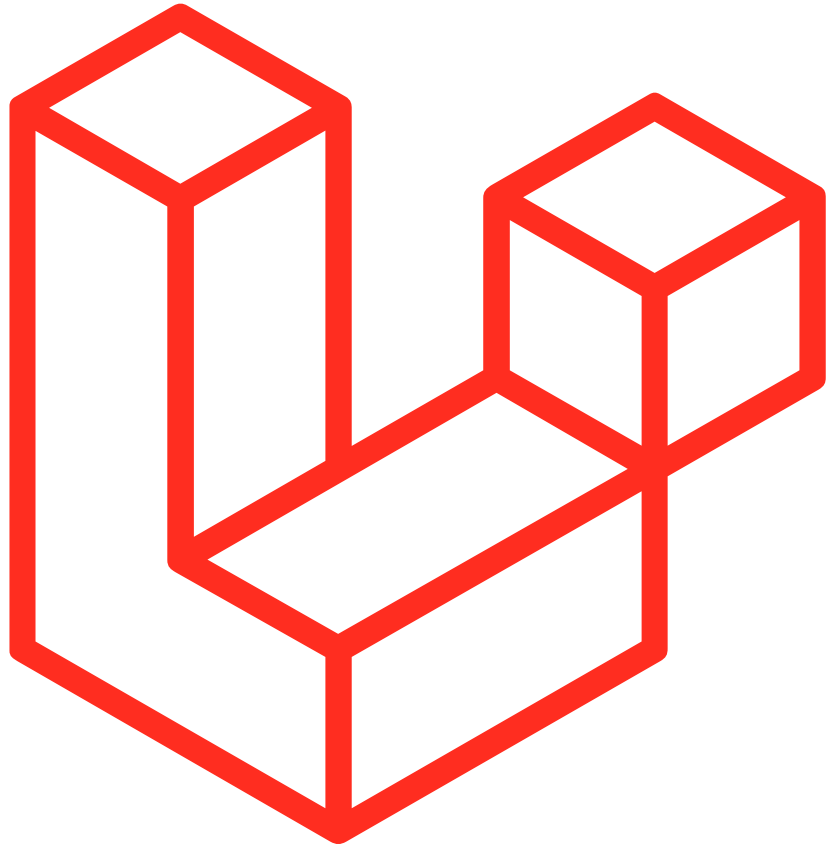
Documentation officielle

- Documentation Laravel : documentation complète.
- Laravel News : actualités et tutoriels.
- Laracasts : vidéos tutoriels.



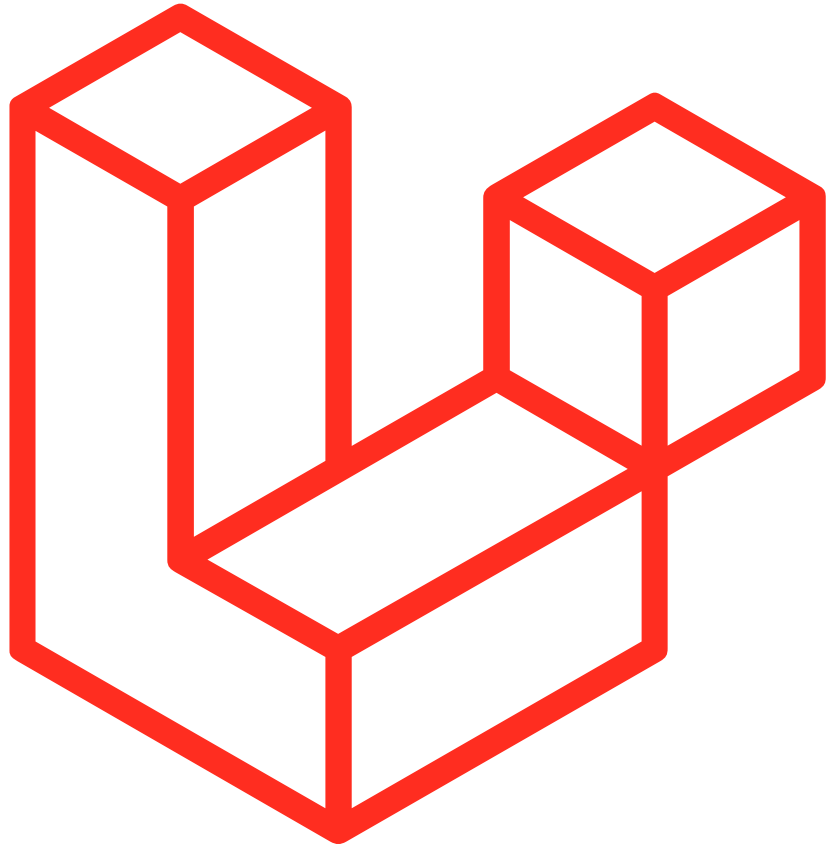
Communauté

- Laravel.io : forum officiel.
- Reddit r/laravel : discussions.
- Discord Laravel : discussions en temps réel.



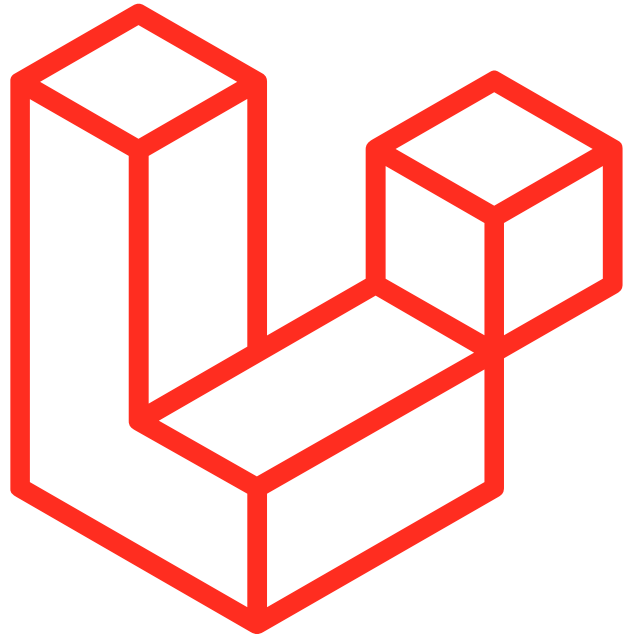
Dépôts GitHub

- laravel/laravel : le dépôt principal du framework.
- Awesome Laravel : liste de ressources Laravel.



Laravel, des briques LEGO parfois un peu magiques

- Laravel peut être intimidant pour les débutant.es.
- Parfois, les fonctionnalités peuvent sembler "magiques".
- Pas besoin de tout comprendre en détail au début.
- Nous allons aborder chaque concept progressivement.



Installation de Laravel

```
# Installe l'installateur Laravel globalement avec Composer
composer global require laravel/installer
```

Créer un nouveau projet Laravel :

```
# Crée un nouveau projet Laravel nommé "mon-projet"
laravel new mon-projet
```

Structure d'un projet Laravel

```
./
├── app/              # Code principal de l'application
├── bootstrap/        # Fichiers de démarrage
├── config/           # Fichiers de configuration
├── database/         # Migrations et seeds
├── node_modules/     # Dépendances JavaScript (si utilisées)
├── public/           # Point d'entrée public (index.php)
├── resources/        # Vues et ressources front-end
├── routes/           # Définitions des routes
├── storage/          # Fichiers générés (logs, cache, etc.)
├── tests/            # Tests unitaires et fonctionnels
└── vendor/           # Dépendances gérées par Composer
```

Lancer le serveur de développement

Laravel inclut un serveur de développement intégré :

```
# Se placer dans le dossier du projet
cd mon-projet

# Lancer le serveur de développement
composer run dev
```

Composer démarrera le serveur de développement et l'application sera alors accessible à l'adresse http://localhost:8000.

PHP vs Composer vs Laravel

PHP vs Composer vs Laravel

Élément	Rôle
PHP	Le langage de programmation
Composer	Le gestionnaire de paquets
Laravel	Le framework

```
PHP (langage)
    └──── Composer (gestionnaire)
              └──── Laravel (framework)
                        └──── Votre application
```

Lien avec JavaScript et npm (1)

Concept	PHP	JavaScript
Gestionnaire de paquets	Composer	npm
Dépôt de paquets	Packagist	npmjs.com
Frameworks web populaires	Laravel, Symfony	Express, React
Fichier de configuration	`composer.json`	`package.json`
Fichier de verrouillage	`composer.lock`	`package-lock.json`
Dossier des dépendances	`vendor/`	`node_modules/`

Lien avec JavaScript et npm (2)

Nous allons éviter de mélanger les deux écosystèmes dans ce cours, mais il est utile de comprendre les parallèles.

Il n'est pas impossible que nous utilisions occasionnellement des outils JavaScript pour le front-end dans les projets Laravel.

Questions

Est-ce que vous avez des questions ?

À vous de jouer !

- (Re)lire le support de cours.
- Faire le mini-projet.
- Faire les exercices.
- Poser des questions si nécessaire.

➡️ Visualiser le contenu complet sur GitHub.

N'hésitez pas à vous entraidez si vous avez des difficultés !

Sources

- Illustration principale par Richard Jacobs sur Unsplash.
- Illustration par Aline de Nadai sur Unsplash.
- Illustration par Heather Zabriskie sur Unsplash.
- Illustration par Lorin Both sur Unsplash.
- Illustration par Nikita Kachanovsky sur Unsplash.